INFORMATION AND CONTROL 19, 439–475 (1971)

Translations on a Context Free Grammar

A. V. AHO AND J. D. ULLMAN*

Bell Telephone Laboratories, Incorporated, Murray Hill, New Jersey 07974

Two schemes for the specification of translations on a context-free grammar are proposed. The first scheme, called a generalized syntax directed translation (GSDT), consists of a context free grammar with a set of semantic rules associated with each production of the grammar. In a GSDT an input word is parsed according to the underlying context free grammar, and at each node of the tree, a finite number of translation strings are computed in terms of the translation strings defined at the descendants of that node. The functional relationship between the length of input and length of output for translations defined by GSDT's is investigated.

The second method for the specification of translations is in terms of tree automata—finite automata with output, walking on derivation trees of a context free grammar. It is shown that tree automata provide an exact characterization for those GSDT's with a linear relationship between input and output length.

I. SYNTAX DIRECTED TRANSLATION

A translation is a set of pairs of strings. One common technique for the specification of translations is to use a context free grammar (CFG) to specify the domain of the translation. An input string is parsed according to this grammar, and the output string associated with this input is specified as a function of the parse tree.

A large class of translations can be specified in this manner. Compilers utilizing this principle are termed syntax directed, and several compilers and compiler writing systems have been built around this concept. See [1–6], for example.

Certain constraints on the source language, such as proper declaration of identifiers or proper use of "go to" statements, cannot be included in the context free specification of the language [7]. However, most of these difficulties can be removed by allowing the use of symbol tables in the

* Current address, Dept. of Electrical Engineering, Princeton University, Princeton, NJ.

implementation. This aspect of compiler writing can also be formalized. (For example, see [8] for a formalism whereby context free grammars are augmented by symbol tables.)

Once the structure of a source program has been "understood" by a compiler, in terms of the context free grammar and symbol tables (i.e., the program has been parsed, and tables have been constructed), the object program can be constructed. The specification for the object program is often made in terms of the parse tree for the source program, and various compiler writing schemes have formalisms for specifying the translation of source code into object code [3–6].

One common formalism for describing translations on a context free grammar is the syntax directed translation scheme (SDT) [2, 9–15]. Here, associated with each production of the underlying CFG is a rule for permuting the order of the nonterminals on the right side of the production and introducing output symbols on the right side. Given a parse tree in the CFG, with a certain production used at some node, the tree is altered at that node by:

(1) deleting descendants with terminal labels,

(2) reordering the nonterminal descendants according to the fixed rule,

and

(3) introducing descendants labeled by output symbols.

The translation of a given input word is thus produced by parsing the input, performing the above operation at each node of the parse tree and taking the yield of the resulting tree as output. (The yield is the string obtained by concatenating the labels of the leaves in order from the left.) If the underlying CFG is ambiguous, several outputs can be defined for one input.

Various generalizations of the SDT have been suggested [11, 14–16]. Of particular interest is the rather general concept of semantics appearing in [16]. Here, after producing a parse tree for the input, an attempt is made to evaluate a set of semantic variables (whose values may be strings, real numbers, list structures or anything else) at each node of the tree. The value of a given variable may depend either on the values of certain variables at its descendants, or on certain variables at its ancestor.

We will here define a class of formal translations that are a generalization of the usual syntax directed translation. Our class can be thought of as restricted semantics in the sense of [16]. The restrictions made are the following:

(1) Semantic variables must take strings as values.

(2) The value of a semantic variable at a given node depends only on the production used at the node and the value of semantic variables at its nonterminal descendants. It is formed from these variables and constant strings by concatenation.

(3) One variable, defined at the root, represents the output.

This scheme, which we call a generalized syntax directed translation (GSDT), is an extension of the T_1 semantics of [14]. The latter are GSDT's with only one semantic variable defined at each node. If we further restrict the GSDT to require that the formula for the variable have exactly one occurrence of the variable at each of its nonterminal descendants, then we have the usual SDT.

The GSDT has features which are not found in the SDT, and which appear useful in practical situations. Consider the context free production

$$\langle\text{for statement}\rangle ::= \textit{for}\ \langle\text{assignment statement}\rangle\ \textit{step}\ \langle\text{integer}\rangle\ \textit{until}\ \langle\text{integer}\rangle\ \textit{do}\ \langle\text{statement}\rangle,$$

which might be used in the specification of some source language.

An example of a $\langle$for statement$\rangle$ is

$$\textit{for}\ I \leftarrow J + 1\ \textit{step}\ 1\ \text{until}\ 20\ \textit{do}\ M \leftarrow M + I.$$

The natural code to be produced from this for statement should do the following:

(1) Compute I (perform the assignment statement).

(2) Test if $I \leqslant 20$ (compare the assigned variable with the second integer).

(3) If not, transfer.

(4) If so, add I to M (execute the statement after *do*).

(5) Increment I by 1 (add the first integer to I).

(6) Return to step (2).

In step (1), we obviously need the code which executes the assignment statement. However, in statements (2) and (5), we need to reference the location reserved for the identifier (I in the example), which can only be determined by examining the assignment statement. Thus, two "translations" of the assignment statement are needed—one which performs the assignment and another which is the location of the identifier whose value is computed. Note that an arbitrarily long sequence of instructions may be required to determine the location, if, for example, the identifier is part of a *PL/I* structure.

It will not quite do to rewrite the production as

⟨for statement⟩ : := *for* ⟨identifier⟩ ← ⟨arith. expression⟩
step ⟨integer⟩ *until* ⟨integer⟩ *do* ⟨statement⟩.

Then, the "translation" of the identifier will have to appear in three portions of the translation, still removing this type of translation from the SDT class.

Let us comment that in a practical system it is useful to have not only string valued variables, but "logical" variables which assume one of a finite number of values and which would determine the rules whereby string variables are computed. For example, consider the translation of arithmetic expressions which are specified by the productions

⟨expression⟩ : := ⟨term⟩ + ⟨expression⟩ | ⟨term⟩
⟨term⟩ : := ⟨factor⟩ * ⟨term⟩ | ⟨factor⟩
⟨factor⟩ : := ⟨identifier⟩ | (⟨expression⟩).

If identifiers can be integer, real or complex, it would be convenient to assign a "semantic attribute", which could have one of these three values, to each expression, factor and term. The attribute would be computed by the expected rules: real + complex = complex, etc. The attributes defined at the descendants of a node influence the interpretation of + and * (for example, whether + should be integer add or floating add) and whether operations such as converting a number from fixed point to floating point should be performed.

While we shall not show it in this paper, we claim that such an extension of the GSDT does not produce any new translations. The proof involves modifying the underlying CFG to incorporate "guesses" as to the value that the logical variables will assume at each node of the parse tree.

What we shall do in this paper is show a necessary condition that a translation be a GSDT. We shall give a restriction on the output length as a function of the number of nodes of the parse tree for any GSDT. This function, broadly speaking is either:

(1) bounded above by a constant,
(2) an integer power of the size of the tree, or
(3) an exponential function of the size of the tree.

If the underlying CFG is unambiguous, "size of the tree" can be replaced by "length of the input." Thus, the translation which takes a string of i 1's to the binary integer i is, unfortunately, not a GSDT, because the output length is the logarithm of the input length. The inverse of this translation is a GSDT, however.

We shall also consider a method of executing translations defined by GSDT's—the tree walking automaton. We shall show that a translation is defined by a tree walking automaton (on the parse trees of some grammar) if and only if it is a GSDT whose output length is a linear function of the input length.

In Section 2, CFG's and parse trees are defined. In Section 3, the GSDT is defined, and the translations generated by GSDT's are characterized in Sections 4 and 5. In Section 6, the tree automaton is defined, and the two concepts are related in Section 7.

II. Context Free Grammars and Parse Trees

A context free grammar (CFG) is a four-tuple $G = (V, \Sigma, P, S)$, where V and Σ are disjoint finite sets of *nonterminals* and *terminals*, respectively. S, in V, is the *start symbol*. P is a finite list of productions of the form $A \to \alpha$, where A is in V and α in $(V \cup \Sigma)^*$.[1] Each production will be assigned an integer index, and the case in which two or more elements of P are identical except for index is not ruled out. The relation $\underset{G}{\Rightarrow}$ is defined on $(V \cup \Sigma)^*$ by: $\alpha A \beta \underset{G}{\Rightarrow} \alpha\gamma\beta$, whenever there is a production $A \to \gamma$ in P. $\overset{*}{\underset{G}{\Rightarrow}}$ is the reflexive, transitive closure of $\underset{G}{\Rightarrow}$. ($\alpha \overset{*}{\underset{G}{\Rightarrow}} \alpha$ for all α; $\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta$ and $\beta \underset{G}{\Rightarrow} \gamma$ implies $\alpha \overset{*}{\underset{G}{\Rightarrow}} \gamma$.)

The language defined by G, denoted $L(G)$, is $\{w \mid w \text{ is in } \Sigma^* \text{ and } S \overset{*}{\underset{G}{\Rightarrow}} w\}$.

A *tree* is a connected directed ordered graph having the following properties:

(1) There is a unique node, called the *root*, which no edge enters.

(2) With the exception of the root, exactly one edge enters each node.

If there is an edge from node N_1 to node N_2, then N_1 is the *ancestor* of N_2, and N_2 is a *descendant* of N_1.

Given a CFG $G = (V, \Sigma, P, S)$, we can define the set of *derivation trees in* G, which are trees with labeled nodes, as follows:

[1] X^* is the set of finite length strings of elements of the set X including ϵ, the string of length 0.

(1) The labels are chosen from $V \cup \Sigma \cup P \cup \{\epsilon\}$.[2]

(2) A single node labeled S is a derivation tree.

(3) Let D be a derivation tree and N a node of D, whose label is A, in V, and which has no descendants. If the i-th production is $A \to X_1X_2 \cdots X_n$, $n \geqslant 1$, each X_j, $1 \leqslant j \leqslant n$, in $V \cup \Sigma$, we can construct a new derivation tree D' by relabeling node N by i and introducing n descendants of N to the tree D. These descendants are labeled $X_1, X_2, \ldots, X_n$, from the left. If the i-th production is $A \to \epsilon$, node N can be given label i and will have a single descendant with label ϵ.

(4) No other trees are derivation trees.

The notion of "to the left of" naturally extends to relate certain nodes which are not the descendants of the same node. That is, if N_1 is to the left of N_2, then all N_1's descendants are to the left of those of N_2.

We call a node a *leaf* if it has no descendants. Note that under our definition of derivation tree, a node is a leaf if and only if its label is in $V \cup \Sigma \cup \{\epsilon\}$. A derivation tree all of whose leaves have terminal or ϵ labels is called a *parse tree*. Given any two leaves, one is to the left of the other. The *yield* of a derivation tree is the string formed by concatenating the labels of the leaves, in order from the left. It is well known that there is a parse tree in grammar G with yield α if and only if $S \overset{*}{\underset{G}{\Rightarrow}} \alpha$.

A derivation *subtree* in grammar G is defined exactly as a derivation tree, except that the label of the root may be any symbol in $V \cup \Sigma \cup P \cup \{\epsilon\}$.

A *path* in a tree is a sequence of nodes $N_1, N_2, \ldots, N_k$, such that N_{i+1} is a descendant of N_i, for $1 \leqslant i < k$. The *length* of this path is $k - 1$. The *height of a node* N is the maximum length of a path $N_1, N_2, \ldots, N_k$, such that $N_1 = N$ and N_k is a leaf. The *height of a tree* is the height of its root.

Let $G_1 = (V_1, \Sigma, P_1, S_1)$ and $G_2 = (V_2, \Sigma, P_2, S_2)$ be two CFG's and h a length preserving homomorphism[3] from V_2 to V_1. We can extend h to $V_2 \cup \Sigma$ by letting $h(a) = a$ for all a in Σ.

Suppose that we can extend h to P_2 in such a manner that if the i-th

[2] We assume the productions are indexed by the integers, and that the integers are not themselves elements of $V \cup \Sigma$. P may have two or more identical elements with distinct indices. Informally, we shall often use the productions rather than the indices as labels, although strictly speaking this could result in confusion if two productions were identical. Most authors use labels from $V \cup \Sigma \cup \{\epsilon\}$ only. However, we find it convenient to identify the production used at each node.

[3] A *homomorphism* h is a single valued map from X to Y^*, for finite sets X and Y. We extend h to domain X^* by letting $h(\epsilon) = \epsilon$ and $h(ua) = h(u)\,h(a)$ for $u \in X^*$, $a \in X$. We say h is *length preserving* if $h(a)$ is a single symbol in Y for all a in X.

production of P_2 is $A \to \alpha$, and $h(i) = j$, then the j-th production of P_1 is $h(A) \to h(\alpha)$. Let D be a parse tree in G_2. We can construct $h(D)$, a parse tree in G_1 with the same yield, by replacing each symbol A in $V \cup \Sigma \cup P \cup \{\epsilon\}$ by $h(A)$. Under such conditions, we say that h is a *tree correspondence* from G_2 to G_1. If, in addition, for every parse tree D' in G_1 there is a unique parse tree D in G_2 such that $h(D) = D'$, then h is a 1-1 *tree correspondence* from G_2 to G_1.

EXAMPLE 2.1. Let

$$G_1 = (\{S, A\}, \{a, b\}, P_1, S) \quad \text{and} \quad G_2 = (\{S, A, B\}, \{a, b\}, P_2, S),$$

where P_1 and P_2 are given by

P_1	P_2
(1) $S \to SS$	(1) $S \to SB$
(2) $S \to aA$	(2) $S \to aA$
(3) $S \to \epsilon$	(3) $S \to \epsilon$
(4) $A \to Sb$	(4) $B \to SB$
	(5) $B \to aA$
	(6) $B \to \epsilon$
	(7) $A \to Sb$

Let $h(A) = A$ and $h(S) = h(B) = S$. (Since there are no duplicate productions, the extension of h to P_2 is now determined.) h is a tree correspondence from G_2 to G_1. Consider the tree of Fig. 1. It is easy to verify

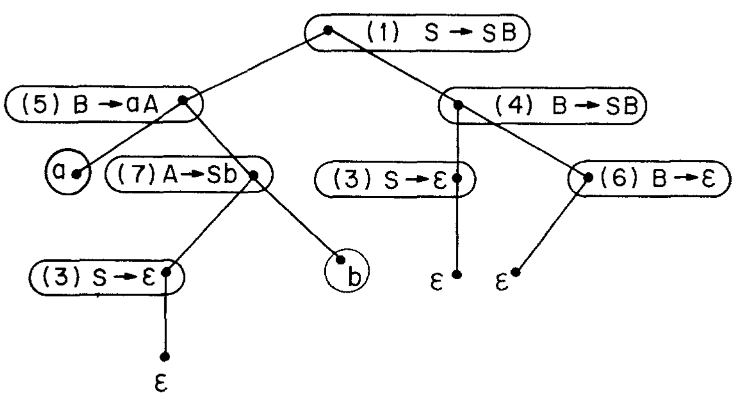

FIG. 1. Tree in grammar G_2.

that if all B's are replaced by S's in Fig. 1, a parse tree in G_1 results. In fact, one can show that h is a 1–1 tree correspondence.

A CFG G is *unambiguous* if for every w in $L(G)$ there is a unique parse tree in G with yield w.[4] The following lemma should be obvious.

LEMMA 2.1. *If there is a* 1–1 *tree correspondence from* CFG G_2 *to* G_1, *then* G_1 *is unambiguous, if and only if* G_2 *is unambiguous.*

A CFG $G = (V, \Sigma, P, S)$ is *proper* if

(1) For all $A \to \alpha$ in P, no nonterminal appears more than once in the string α.

(2) S appears on the right of no production.

(3) For all A in V, $S \overset{*}{\underset{G}{\Rightarrow}} w_1 A w_2$ and $A \overset{*}{\underset{G}{\Rightarrow}} w_3$, for some terminal strings w_1, w_2 and w_3.

The following lemma is elementary, and the proof is omitted.

LEMMA 2.2. *Given a* CFG G_1, *one can find an equivalent*[5] *proper* CFG G_2 *and a* 1–1 *tree correspondence from* G_2 *to* G_1.

From here on, we assume a CFG to be proper. All CFG's constructed will have that property. The restriction of "properness" is made to simplify the description of a GSDT, and using Lemma 2.2, one can easily show the restriction to be without loss of generality as far as the defining power of the GSDT or any other property of GSDT's discussed here is concerned.

III. GENERALIZED SYNTAX DIRECTED TRANSLATIONS

A generalized syntax directed translation (GSDT) is a four-tuple $F = (G, \Delta, \Gamma, R)$, where:

(1) $G = (V, \Sigma, P, S)$ is a proper context free grammar;

(2) Δ is a finite set of *output symbols*;

(3) Γ is a finite set of distinct *translation symbols* of the form $\tau_i(A)$, where i is an integer and A is in $V - \{S\}$, plus the symbol S_1. Whenever it is possible to do so without confusion, we will denote $\tau_i(A)$ by A_i. We call A_i the i-th translation symbol associated with A.

(4) R is a function which associates with each production $A \to \alpha$ in P,

[4] Note that any grammar with two identical productions is ambiguous in our sense.

[5] G_1 is equivalent to G_2 if $L(G_1) = L(G_2)$.

a set of *semantic rules* $\{A_1 = \beta_1, A_2 = \beta_2, ..., A_m = \beta_m\}$, in which each β_i is a string in $(\Gamma \cup \Delta)^*$, such that all translation symbols appearing in β_i are translation symbols associated with nonterminals appearing in α.

For each x in Σ^* we define $F(x)$, the set of outputs of x as follows:

(1) If x is not in $L(G)$, then $F(x) = \varphi$.

(2) If x is in $L(G)$, then each parse tree with yield x defines an element y in $F(x)$, which is the value of the translation symbol S_1 associated with the root. The value of S_1 is computed bottom-up as follows:

(i) With each interior node N of the parse tree labeled $A \to \alpha$ are associated the translation symbols $A_1, A_2, ..., A_m$, which are all the translation symbols associated with A. The values of these translation symbols at N are computed using the semantic rules and the values of the translation symbols at the descendants of N as follows.

(ii) Suppose α is $x_0B_1x_1B_2x_2 \cdots B_kx_k$, where x_j is in Σ^* and B_j is in V, $0 \leqslant j \leqslant k$. Suppose $A_i = y_0C_1y_1C_2y_2 \cdots C_ly_l$ is the semantic rule for A_i, where y is in Δ^* and C_j is a translation symbol in Γ associated with B_{h_j} for some $1 \leqslant h_j \leqslant k$. Then $v(A_i)$, the *value* of A_i at node N, is the string $y_0v(C_1)y_1v(C_2)y_2 \cdots v(C_l)y_l$ in Δ^*, where $v(C_j)$ is the value of C_j at the descendant of N which is labeled by a B_{h_j} production.[6]

$T(F)$, the *translation defined by F*, is the set $\{(x, y) \mid y \in F(x)\}$.

EXAMPLE 3.1. Let $F = (G, \{a, b\}, \{S_1, A_1, A_2, B_1, B_2\}, R)$, where the productions of the grammar and the associated semantic rules are:

	Productions	Semantic rules
(1)	$S \to A$	$S_1 = A_1A_2$
(2)	$A \to aAbB$	$A_1 = aA_1B_1$
		$A_2 = bA_2B_2$
(3)	$A \to bAaB$	$A_1 = aA_1B_1$
		$A_2 = bA_2B_2$
(4)	$B \to A$	$B_1 = A_1$
		$B_2 = A_2$
(5)	$A \to \epsilon$	$A_1 = \epsilon$
		$A_2 = \epsilon$

F defines the translation $\{(w, a^ib^i) \mid i \geqslant 0$ and $w \in \{a, b\}^*$, such that w has s and $s\}$. Intuitively, the translations A_1 and B_1 accumulate a's; A_2 and B_2

[6] Note that the properness of G makes this descendant unique.

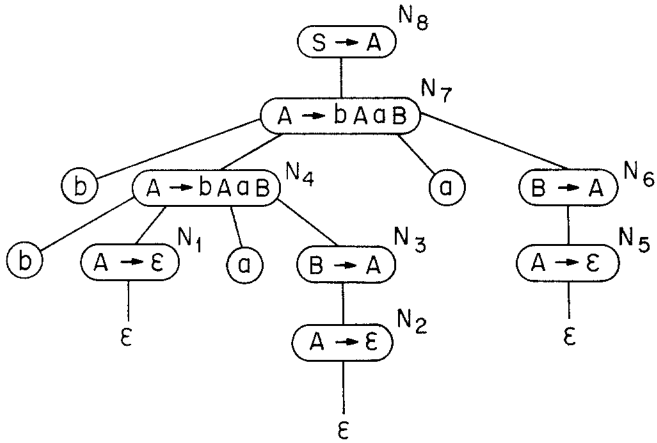

FIG. 2. Parse tree for *bbaa*.

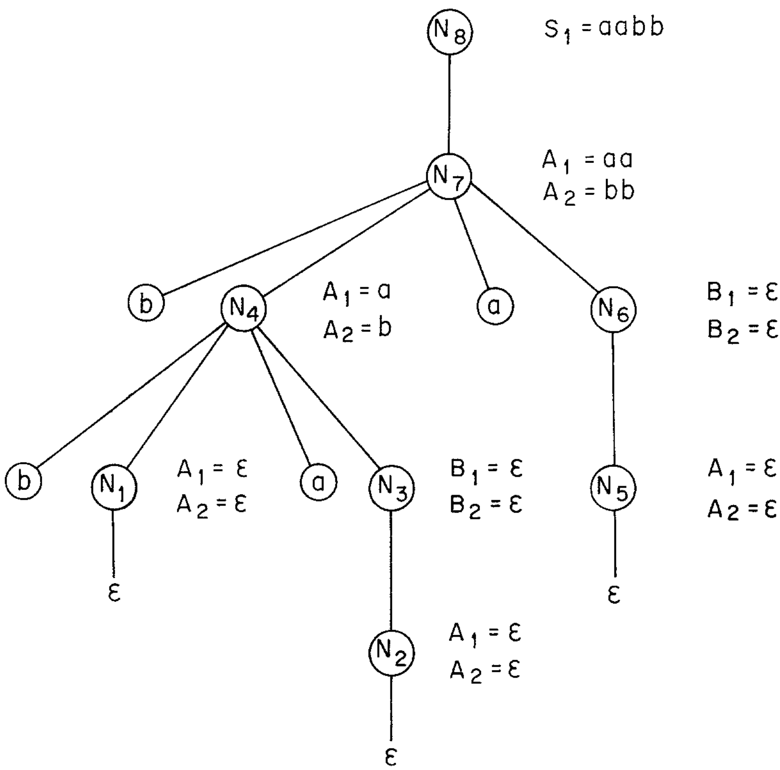

FIG. 3. Tree with values of translation symbols.

accumulate b's. Note that A could not be substituted for B in productions (2) and (3) without violating the properness condition. For example, consider the input word *bbaa*. The parse tree for *bbaa* is shown in Fig. 2. We have numbered the interior nodes for convenience. The translation symbols A_1 and A_2 are associated with N_1, and $v(A_1) = v(A_2) = \epsilon$ at N_1. Translation symbols A_1 and A_2 are also associated with N_2, and $v(A_1) = v(A_2) = \epsilon$ at N_2. Translation symbols B_1 and B_2 are associated with N_3, and $v(B_1)$ and $v(B_2)$ at N_3 are equal to $v(A_1)$ and $v(A_2)$, respectively, at N_2. Thus $v(B_1) = v(B_2) = \epsilon$ at N_3. N_4 has translation symbols A_1 and A_2, and $v(A_1) = a$ and $v(A_2) = b$ at N_4.

The values of the associated translation symbols at each node are shown in Fig. 3. Since the value of S_1 at the root is *aabb*, *aabb* is in $F(bbaa)$.

EXAMPLE 3.2. Let $F = (G, \{b\}, \{S_1, A_1\}, R)$ where the productions and associated semantic rules are:

	Productions	Semantic rules
(1)	$S \to aA$	$S_1 = bA_1A_1$
(2)	$S \to aA$	$S_1 = A_1A_1$
(3)	$A \to aA$	$A_1 = bA_1A_1$
(4)	$A \to aA$	$A_1 = A_1A_1$
(5)	$A \to \epsilon$	$A_1 = \epsilon$

This GSDT is an example of the use of identical productions. It maps

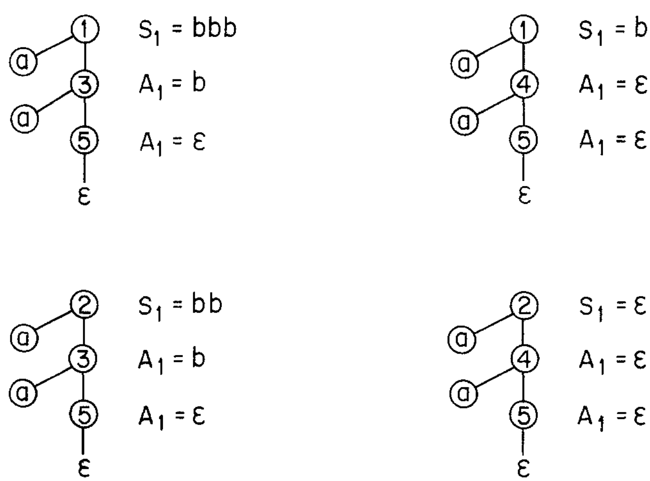

FIG. 4. Parse trees of *aa*.

a^i to all strings b^j such that $0 \leqslant j < 2^i$. The input aa has four parse trees, shown together with the values of the translation symbols computed at each node, in Fig. 4.

A translation defined by a GSDT $F = (G, \Delta, \Gamma, R)$ is said to be *unambiguous* if the underlying CFG G is unambiguous. An unambiguous translation has the important property that there exists exactly one translation for each input word in $L(G)$. However, an ambiguous translation need not have more than one output for each input word. As an example, the grammar in the GSDT of Example 3.1 is ambiguous, but for each x in $\{a, b\}^*$, $F(x)$ contains at most one element.

IV. Proliferation of Translations

Because of the manner in which the output of a parse tree is computed, the value of a particular translation symbol A_i at a given node of the parse tree can appear many times in the value of S_1 at the root. The function relating the maximum number of times any translation of A can appear in the output, taken over all nonterminals A, as a function of the number of nodes in the parse tree is termed the proliferation rate of S_1.

In this section we shall show that the proliferation rate of any translation symbol of a GSDT is either an integer power of n (possibly zero) or exponential in n, where n is the number of nodes in the parse tree.

As an example, suppose that a GSDT contains the following productions and associated semantic rules.

Production	Semantic rules
$A \to BC$	$A_1 = B_1C_1B_2$
$B \to DE$	$B_1 = D_1E_1$
	$B_2 = D_1D_2$
$D \to \alpha$	$D_1 = \beta_1$
	$D_2 = \beta_2$

If the structure of Fig. 5 appears in a parse tree, then the value of A_1 at the node labeled N_1 will involve the values of B_1 and B_2 at the node labeled N_2. Thus, the value of A_1 at N_1 has two substrings, both of which can be regarded as translations of the input string derived from the node labeled N_2. Similarly, the value of A_1 at N_1 involves three translations of the string derived from the node labeled N_3; two of these substrings are the value of D_1 at N_3 and one is the value of D_2 at N_3. For large parse trees, the value of the translation symbols at one node may be reproduced many times at another node.

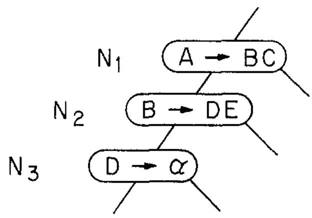

FIG. 5. Portion of a tree.

To investigate this matter, we define the notion of proliferation rate for translation symbols of a GSDT $F = (G, \Delta, \Gamma, R)$, where $G = (V, \Sigma, P, S)$, as follows. Define $\mathscr{H}_F$ to be the set of homomorphisms h_{iB} from Γ to Γ^*, where i is the number of a production, say $A \to \alpha$, and B is a nonterminal in V. If R associates $A_j = \beta$ with production i, then $h_{iB}(A_j)$ is the string obtained from β by deleting those symbols which are not translation symbols associated with B. We let $h_{iB}(C_k) = \epsilon$ if $C \neq A$.

EXAMPLE 4.1. Consider the following GSDT F with productions and rules as shown.

(1)	$S \to 1A$	$S_1 = A_1A_2$
(2)	$A \to 1A$	$A_1 = A_1A_2$
		$A_2 = A_2A_2$
(3)	$A \to 0A$	$A_1 = A_1$
		$A_2 = A_2A_2$
(4)	$A \to G$	$A_1 = \epsilon$
		$A_2 = a$

The reader can verify that $T(F) = \{(x, y) \mid x$ is the binary representation of n, $n \geqslant 1$ and $y = a^n\}$. Here $\mathscr{H}_F$ is the set $\{h_{iB} \mid 1 \leqslant i \leqslant 4,\ B$ is A or $S\}$. These homomorphisms are defined by:

(1) $h_{1A}(S_1) = h_{2A}(A_1) = A_1A_2$
(2) $h_{2A}(A_2) = h_{3A}(A_2) = A_2A_2$
(3) $h_{3A}(A_1) = A_1$
(4) $h_{iB}(X) = \epsilon$ otherwise.

We will use these homomorphisms to define the way in which the value of a particular translation can depend on the values of translations at nodes far removed from it. We can consider the composition of homomorphisms in $\mathscr{H}_F$, and represent these by strings in $\mathscr{H}_F^*$, with the rightmost symbol to be applied first.[7] If C is in Γ, we define the *proliferation rate of* C, denoted $f_C(n)$, to be $\max_{\alpha \text{ in } (\mathscr{H}_F)^n} | \alpha(C) |$.[8]

Observe that a string of homomorphisms $\alpha = h_{i_m B_m} \cdots h_{i_2 B_2} h_{i_1 B_1}$ [9] represents a path of length m in a derivation tree, provided that for $1 \leqslant j < m$, i_{j+1} is a B_j-production and, for all j, production i_j has an instance of B_j on the right. The labels of the first m nodes in this path are $i_1, i_2, \ldots, i_m$, and the label of the last node in this path is either B_m or a B_m-production. The choice of descendant from each node is indicated by $B_1, B_2, \ldots, B_m$. The path α is sketched in Fig. 6.

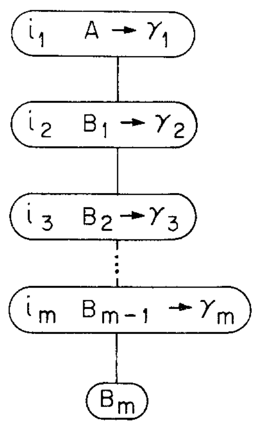

FIG. 6. Path in parse tree.

If production i_1 is an A-production, and the symbol $\tau_k(B_m)$ appears p times in the string $\alpha(\tau_j(A))$, then p copies of the value of $\tau_k(B_m)$ will be included in the value of $\tau_j(A)$ at the node labeled i_1. (It is straightforward to show this by induction on m.) Thus, the proliferation rate of a translation symbol A_j is the maximum number of translations of any nonterminal B which can appear in the value of A_j at a node N as a function of the length of the path from N to the node labeled by B or a B-production.

In the next section we will use the proliferation rate of the translation

[7] Conventionally, we take the empty string of homomorphisms to be the identity homomorphism.

[8] $| x |$ denotes the length of x, (i.e., the number of symbols in x).

[9] Note that subscripted capital letters here represent a sequence of nonterminal, not translation, symbols.

symbols of a GSDT to determine the growth of the output length of the translation defined by the GSDT as a function of the number of nodes of a parse tree.

EXAMPLE 4.2. Consider the GSDT in Example 4.1. Consider the path $h_{2A}^{n-1}h_{1A}$ in a parse tree. Since $h_{2A}^{n-1}h_{1A}(S_1) = A_1A_2^{2^n-1}$, the proliferation rate of S_1 is easily seen to be $f_{S_1}(n) = 2^n$.

Let $F = (G, \Delta, \Gamma, R)$ be a GSDT. We define sets $\Gamma^{(i)} \subseteq \Gamma$ and $\Gamma^{[j]}$, for integers $i \geqslant 0, j \geqslant -1$, by

(1) $\Gamma^{[-1]} = \varphi$.

(2) For $i \geqslant 0$, $\Gamma^{(i)}$ is the set of C in $\Gamma - \Gamma^{[i-1]}$ such that for some constant c, $f_C(n) \leqslant cn^i$ for all $n \geqslant 1$.

(3) For $i \geqslant 0$, $\Gamma^{[i]} = \Gamma^{[i-1]} \cup \Gamma^{(i)}$.

That is, $\Gamma^{(i)}$ is the set of C in Γ such that $f_C(n)$ is greater than cn^{i-1} but at most cn^i. We shall show that if C is in $\Gamma^{(i)}$, $i \geqslant 1$, then $f_C(n)$ is proportional to n^i.

LEMMA 4.1. *Let $F = (G, \Delta, \Gamma, R)$, let α be in $\mathscr{H}_F{}^*$ and C be in $\Gamma^{(i)}$. Then:*

(a) *If $\alpha(C)$ has an instance of symbol D, then D is in $\Gamma^{[i]}$.*

(b) *If $\alpha(C)$ has an instance of a symbol in $\Gamma^{(j)}$, and $\alpha = \beta\gamma$, then $\gamma(C)$ has a symbol in $\Gamma^{(k)}$ for some $k \geqslant j$.*

Proof. (a) If not, then we can easily find continuations $\beta_1, \beta_2, \ldots$ of α, such that the number of symbols in $\beta_j\alpha(C)$ is greater than $c \mid \beta_j\alpha \mid^i$ for any fixed c and arbitrary j. (b) If $\gamma(C)$ is in $(\Gamma^{[j-1]})^*$, a violation of (a) would occur.

We next show a "pumping lemma" for translation symbols.

LEMMA 4.2. *Let $F = (G, \Delta, \Gamma, R)$, Γ' be a subset of Γ, and C be in Γ. If there is no constant upper bound on the number of instances of a symbol of Γ' in $\alpha(C)$ for α in $\mathscr{H}_F{}^*$, then there exist β_1, β_2 and β_3 in $\mathscr{H}_F{}^*$ such that for all m, $\beta_1\beta_2{}^m\beta_3(C)$ has at least $m + 1$ instances of symbols in Γ'.*

Proof. Let Γ have s symbols and let r be the maximum of $| h(X)|$ for X in Γ and h in $\mathscr{H}_F$. By hypothesis, there is some α in $\mathscr{H}_F{}^*$ such that $\alpha(C)$ has more than r^{2^s} instances of symbols in Γ'. Let $\alpha = g_k \cdots g_2 g_1$, with g's in $\mathscr{H}_F$, and define $\alpha_i = g_i \cdots g_2 g_1$ for $0 \leqslant i \leqslant k$. Define Γ_i to be the set of D appearing in $\alpha_i(C)$, such that $g_k g_{k-1} \cdots g_{i+1}(D)$ has at least one element of Γ'. Note that $\Gamma_i \neq \varphi$.

Since an element of $\mathscr{H}_F$ increases the length of a string upon which it operates by at most a factor of r, we can find a set Q of $2^s + 1$ integers between 0 and k, such that if i and j are in Q, $i < j$, then $\alpha_j(C)$ has more instances of symbols in Γ_j than $\alpha_i(C)$ has instances of symbols in Γ_i. Thus, we can find i and j in Q, with $\Gamma_i = \Gamma_j = \hat{\Gamma}$ and $j > i$. Observe that if D is in $\hat{\Gamma}$, then $g_j g_{j-1} \cdots g_{i+1}(D)$ contains at least one instance of an element of $\hat{\Gamma}$, and for some D' in $\hat{\Gamma}$, $g_j g_{j-1} \cdots g_{i+1}(D')$ contains at least two such instances. Let $\beta_1 = g_k g_{k-1} \cdots g_{j+1}$, $\beta_2 = g_j g_{j-1} \cdots g_{i+1}$ and $\beta_3 = g_i g_{i-1} \cdots g_1$. The lemma follows immediately.

The next two lemmas give a recursive criterion for determining the members of $\Gamma^{(i)}$, $i \geqslant 0$.

Lemma 4.3. *Let* $F = (G, \Delta, \Gamma, R)$ *be a* GSDT, *and suppose* C *is in* $\Gamma - \Gamma^{[i]}$, $i \geqslant -1$. *If there is a constant* c *such that for all* α *in* $\mathscr{H}_F{}^*$, $\alpha(C)$ *has at most* c *instances of a symbol not in* $\Gamma^{[i]}$, *then* C *is in* $\Gamma^{(i+1)}$.

Proof. The case $i = -1$ is by definition. Assume $i \geqslant 0$. Let $\alpha = g_n \cdots g_2 g_1$ be in $\mathscr{H}_F{}^*$. Define $w_j = g_j \cdots g_2 g_1(C)$, for $0 \leqslant j \leqslant n$. Let x_j be the string obtained by deleting all occurrences of symbols in $\Gamma^{[i]}$ from w_j. Let y_{j+1} be the string consisting of those symbols of $g_{j+1}(x_j)$ which are in $\Gamma^{[i]}$. It follows by Lemma 4.1 and induction on j that w_j is a permutation of the symbols of the strings x_j, y_j, $g_j(y_{j-1})$, $g_j g_{j-1}(y_{j-2}),\ldots, g_j g_{j-1} \cdots g_2(y_1)$.

By hypothesis, $|x_k| \leqslant c$ for all k. Let r be the maximum of $|h(D)|$ for D in Γ, h in $\mathscr{H}_F$. Then $|y_k| \leqslant cr$ for all k. Since y_k consists only of symbols in $\Gamma^{[i]}$, there is a constant c', depending only on F, such that

$$|g_n g_{n-1} \cdots g_k(y_{k-1})| \leqslant cc'r(n - k + 1)^i.$$

Thus, $|w_n| \leqslant c + cr + cc'r[1^i + 2^i + \cdots + (n-1)^i] \leqslant c''n^{i+1}$ for some constant c''.

Lemma 4.4. *Let* $F = (G, \Delta, \Gamma, R)$ *and let* C *be in* $\Gamma - \Gamma^{[i]}$, $i \geqslant 0$. *Then there is a constant* $c > 0$ *such that for all* n, *there is an* α_n *in* $(\mathscr{H}_F)^n$, *such that* $\alpha_n(C)$ *has at least* cn *instances of symbols of* $\Gamma - \Gamma^{[i-1]}$.

Proof. By Lemma 4.3, there is no constant upper bound on the number of instances of symbols in $\Gamma - \Gamma^{[i-1]}$ found in $\alpha(C)$, for α in $\mathscr{H}_F{}^*$. By Lemma 4.2, there exist β_1, β_2 and β_3 in $\mathscr{H}_F{}^*$ such that $\beta_1\beta_2{}^j\beta_3(C)$ has at least $j+1$ symbols in $\Gamma - \Gamma^{[i-1]}$. Let r be the maximum length of the right side of a rule of F.

As a general observation, if γ_1 and γ_2 are in $\mathscr{H}_F{}^*$, and $\gamma_1\gamma_2(C)$ has k instances of symbols in $\Gamma - \Gamma^{[i-1]}$, then $\gamma_2(C)$ has a least $k/r^{|\gamma_1|}$ instances of symbols from $\Gamma - \Gamma^{[i-1]}$. (This follows directly from Lemma 4.1.) Choose j_n to be the smallest integer $j \geqslant 0$ such that $|\beta_1\beta_2{}^j\beta_3| \geqslant n$, and let α_n be the rightmost n symbols of $\beta_1\beta_2^{j_n}\beta_3$. By the minimality of j_n, we know that $|\beta_1\beta_2^{j_n}\beta_3| - |\alpha_n| \leqslant |\beta_1\beta_2\beta_3|$. Thus, $\alpha_n(C)$ has at least $(j_n + 1)/r^{|\beta_1\beta_2\beta_3|}$ instances of a symbol in $\Gamma - \Gamma^{[i-1]}$. Since $j_n \geqslant [n/|\beta_1\beta_2\beta_3|]$,[10] we have $j_n + 1 \geqslant n/|\beta_1\beta_2\beta_3|$. Thus, the lemma is satisfied with $c = 1/r^{|\beta_1\beta_2\beta_3|}\,|\beta_1\beta_2\beta_3|$.

We now prove that there are no proliferation rates between n^i and n^{i+1} for integer i.

LEMMA 4.5. *Let $F = (G, \Delta, \Gamma, R)$, and let C be in $\Gamma - \Gamma^{[i]}$, $i \geqslant 0$. Then there is a constant $c > 0$, such that $f_C(n) \geqslant cn^{i+1}$, for all n.*

Proof. The result is immediate from Lemma 4.4 for $i = 0$. Assume it true for $i < j$ and let C be in $\Gamma - \Gamma^{[j]}$. By Lemma 4.4, there is a constant c_1 such that for all n there exists γ_n in $(\mathscr{H}_F)^n$ such that $\gamma_n(C)$ has at least $c_1 n$ instances of symbols of $\Gamma - \Gamma^{[j-1]}$. By the inductive hypothesis, there is a constant c_2 such that for all n and any D in $\Gamma - \Gamma^{[j-1]}$, there is a β_{nD} in $(\mathscr{H}_F)^n$ for which $|\beta_{nD}(D)| \geqslant c_2 n^j$.

Let Γ have s symbols and let r be the maximum length of the right side of a rule. Let D_n be an element of $\Gamma - \Gamma^{[j-1]}$ such that $\gamma_n(C)$ has at least $c_1 n/s$ instances of D_n. Then $|\beta_{nD_n}\gamma_n(C)| \geqslant (c_2 n^j)(c_1 n)/s$. Define α_{2n} to be $\beta_{nD_n}\gamma_n$ and α_{2n-1} to be α_{2n} with the leftmost symbol deleted. Then for all m, $|\alpha_m(C)| \geqslant (c_2(m/2)^j)(c_1(m/2))/rs$. The lemma then follows with $c = c_1c_2/rs2^{j+1}$.

From Lemma 4.5, we immediately have:

THEOREM 4.1. *Let $F = (G, \Delta, \Gamma, R)$ be a* GSDT. *Then for any $i \geqslant 1$ and all C in $\Gamma^{(i)}$, there are positive constants c_1 and c_2 such that $c_1 n^i \leqslant f_C(n) \leqslant c_2 n^i$.*

THEOREM 4.2. *Let $F = (G, \Delta, \Gamma, R)$ and let C be in Γ. C is in $\Gamma^{(i)}$, $i \geqslant 0$, if and only if C is not in $\Gamma^{[i-1]}$, and there is a constant c such that $\alpha(C)$ has no more than c instances of elements of $\Gamma - \Gamma^{[i-1]}$, for any α in $\mathscr{H}_F{}^*$.*

Proof. The theorem is true by definition for $i = 0$. For $i \geqslant 1$, the "if" portion is Lemma 4.3. For the "only if" part, assume C is in $\Gamma - \Gamma^{[i-1]}$, and there is no bound on the number of symbols in $\Gamma - \Gamma^{[i-1]}$ possessed by any $\alpha(C)$. By Lemma 4.2 there exist β_1, β_2 and β_3 in $\mathscr{H}_F{}^*$, such that $\beta_1\beta_2{}^j\beta_3(C)$ has at least $j + 1$ instances of a symbol in $\Gamma - \Gamma^{[i-1]}$. Using Lemma 4.5, it is then easy to show that C is not in $\Gamma^{(i)}$.

[10] $[x]$ is the integer part of x.

THEOREM 4.3. *It is decidable if C is in $\Gamma^{(i)}$ for any C in Γ and $i \geqslant 0$.*

Proof. In the proof of Lemma 4.2, a finite test to determine whether C can generate strings with an arbitrary number of symbols in $\Gamma - \Gamma^{[i-1]}$ is implied.

THEOREM 4.4. *Let $F = (G, \Delta, \Gamma, R)$. If C is in $\Gamma^{(i)}$, $i \geqslant 1$, then there is a constant c such that for all n, there exists α_n in $\mathscr{H}_F{}^*$ for which $\alpha_n(C)$ has at least cn instances of an element of $\Gamma^{(i-1)}$.*

Proof. By Lemma 4.4 there is a constant c_1 and some β_n in $(\mathscr{H}_F)^n$ for each n, such that $\beta_n(C)$ has at least $c_1 n$ instances of an element of $\Gamma - \Gamma^{[i-2]}$. By Lemma 4.1, $\beta_n(C)$ has $c_1 n$ instances of an element of $\Gamma^{(i-1)} \cup \Gamma^{(i)}$. But by Theorem 4.2, there is a constant c_2 such that at most c_2 of these instances are in $\Gamma^{(i)}$. The theorem then follows by algebraic manipulation.

THEOREM 4.5. *For every* GSDT *$F = (G, \Delta, \Gamma, R)$ there is an integer $i \geqslant -1$ such that $\Gamma^{(j)}$ is nonempty if and only if $0 \leqslant j \leqslant i$.*

Proof. Immediate from Theorem 4.4.

THEOREM 4.6. *Let $F = (G, \Delta, \Gamma, R)$. If C in Γ is not in $\Gamma^{(i)}$ for any value of i, then there are positive constants c and k such that $f_C(n) \geqslant kc^n$.*

Proof. Let Γ' be the set of D in Γ which are in no $\Gamma^{(i)}$. By Theorems 4.2 and 4.5, for each D in Γ' there is some C in Γ' and α_D in $\mathscr{H}_F{}^*$, such that $\alpha_D(D)$ has at least two instances of C. Let c_1 be the maximum length of α_D for any D in Γ', and let r be the maximum length of the right side of a rule of F. Then for any n, we can find β_n in $(\mathscr{H}_F)^n$ such that $|\beta_n(C)| \geqslant r^{-c_1} 2^{n/c_1}$. Let $k = r^{-c_1}$ and $c = 2^{1/c_1}$.

V. INPUT–OUTPUT LENGTH RELATIONSHIPS

One quantity which is of particular practical interest in the definition of translations is how the length of the output varies as the length of the input. In particular, we might like to know that inputs cannot give outputs too much longer than themselves. Investigation of this matter yields a necessary condition that a translation be a GSDT. We define the *output growth* of a translation T as the function $g(n) = \max_{|x|=n} \min_{(x,y) \text{in} T} |y|$. $g(n)$ will be undefined for those values of n for which there is no word of length n in the domain of T.

There is a close relationship between the output growth of a translation defined by a GSDT and the proliferation rate of its translation symbols. Informally speaking, we shall show that if T is an unambiguous translation defined by some GSDT F', then T can be defined by a GSDT F such that if the proliferation rate of the translation symbol S_1 of F is $f(n)$, then $T(F)$ has output growth proportional to $nf(n)$.

For ambiguous translations defined by GSDT's this relation between output growth and growth rate of translation symbols is somewhat obscured when there is more than one output word for an input word. However, in general we can show that if a translation T is defined by some GSDT F', and T has an infinite domain, then T is defined by a GSDT F such that if the proliferation rate of S_1 is $f(n)$, then there exists an infinity of x in the domain of T such that (x, y) is in T and $|y|$ is proportional to $nf(|x|)$. Moreover, the output growth of T is at most proportional to $nf(n)$. For this purpose, the following concepts will be useful.

If x in Δ^* is a translation string defined at some node of a parse tree, $|x| = n$, then we say that x has *n positions*, each containing one of the symbols of string x. The positions are numbered $1, 2,..., n$, from the left.

Let $v(A_i)$, the value of A_i, be computed at some node N by the rule $A_i = B_1B_2 \cdots B_m$. We assign an *origin* to each position of $v(A_i)$ at N as follows:

(1) Suppose we have assigned an origin to the first j positions of $v(A_i)$, $j \geqslant 0$, and have considered $B_1, B_2,..., B_k$, $0 \leqslant k < m$.

(2) If B_{k+1} is in Δ, then position $j + 1$ of $v(A_i)$ is said to be an *introduced position*, and B_{k+1} an *introduced symbol*. The origin of position $j + 1$ is the $k + 1$st position of the rule $A_i = B_1B_2 \cdots B_m$. We have now assigned an origin to the first $j + 1$ positions of $v(A_i)$ and considered B_{k+1}.

(3) Suppose $B_{k+1} = C_l$ is in Γ, and the string $v(C_l)$ is defined at the descendant of node N labeled by a C-production. If $|v(C_l)| = m$, then the origin of position $j + p$ of $v(A_i)$ is the p-th position of $v(C_l)$, $1 \leqslant p \leqslant m$. We have now assigned an origin to the first $j + m$ symbols of $v(A_i)$ and considered B_{k+1}.

We extend the notion of origin transitively and reflexively. That is, any position is its own origin, and if a position p_1 in some string is an origin of position p_2 in a second string, and p_2 is in turn an origin of a position p_3 in some third string, then p_1 is an origin of p_3.

Informally, we can visualize the origin relation as follows. At some node N of a parse tree, mark some position p of the value of a translation defined at N, by changing the symbol there to some new symbol. Then, recompute the translation strings at the nodes above N. The positions of the various strings which hold the new symbol are those of which position p is an origin.

Let N_1 and N_2 be two nodes of a parse tree, with a path from N_1 to N_2. Let p be a position of a string $v(B_j)$ defined at N_2 and let $v(A_i)$ be defined at N_1. The *multiplicity* of position p in $v(A_i)$ is the number of positions of $v(A_i)$ having p as origin. This set of positions is called the *projection* of p to $v(A_i)$.

Some basic relations are stated without proof.

LEMMA 5.1. (a) *If p_1 and p_2 are two distinct positions of translation strings at a node N_1, then the projections of p_1 and p_2 to any string defined at any node are disjoint.*

(b) *Every position of every translation string has a unique origin (possibly itself) which is an introduced symbol.*

(c) *Let N_1 and N_2 be nodes of a parse tree, with a path from N_1 to N_2. Let α in $\mathscr{H}_F^*$ represent that path. If $v(A_i)$ is defined at N_1 and $v(B_j)$ at N_2, then the multiplicity of any position of $v(B_j)$ at N_2 in $v(A_i)$ at N_1 equals the number of instances of the symbol B_j in $\alpha(A_i)$.*

We now proceed to relate the proliferation rate in translation symbols to the output growth of translations.

LEMMA 5.2. *Let $F = ((V, \Sigma, P, S), \Delta, \Gamma, R)$ be a* GSDT, *and suppose that there are constants c_1 and c_2 such that the proliferation rate of S_1 is bounded above by $c_1(n+1)^{c_2}$. Then there is a constant c_3, such that if $v(S_1)$ is defined at the root of a parse tree D with n nodes, $|v(S_1)| \leqslant c_3 n^{c_2+1}$.*

Proof. Let r be the maximum length of the right side of a semantic rule. By Lemma 5.1(b), $|v(S_1)|$ at the root of D is the sum over each introduced symbol (in the value of each translation symbol at each node of D) of the multiplicity of that symbol in $v(S_1)$. By Lemma 5.1(c), this sum is bounded by r times the number of nodes of D times the maximum multiplicity in $v(S_1)$ of a position in the value of a translation symbol. No path in D has length greater than $n - 1$. Thus, $|v(S_1)| \leqslant rnc_1n^{c_2}$ and the lemma is proven, with $c_3 = c_1 r$.

LEMMA 5.3. *If F is as in Lemma 5.2, but the growth rate of S_1 is bounded above by c^n for some constant c, then there is a constant c' such that for a tree with n nodes, $v(S_1)$ at the root has length at most $(c')^n$.*

Proof. Similar to Lemma 5.2.

We will now give a general framework for the modification of GSDT's in such a way that some desired information is carried along at the nodes of parse trees. Let $G = (V, \Sigma, P, S)$ be a proper CFG, and let Y be a finite set of symbols. Let μ be a mapping from $P \times Y$ to the finite subsets of Y^* such that:

(1) if $\mu(i, A)$ contains w, $|w| = k$, then the i-th production of P has k nonterminals on the right, and

(2) if the i-th production has k nonterminals on the right, then every w in Y^k is in $\mu(i, A)$ for exactly one value of A.

Given Y and μ, we can construct from G an equivalent CFG G', such that the nodes of a parse tree in G' contain an additional finite amount of information represented by the elements of Y. Specifically, let G' be the CFG (V', Σ, P', S) where $V' = \{S\} \cup [(V - \{S\}) \times Y]$ and P' be defined as follows.

(a) Let the i-th production in P be $S \to A_1 A_2 \cdots A_n$. Then all productions $S \to B_1 B_2 \cdots B_n$ are in P', where $B_j = A_j$ whenever A_j is in Σ, and $B_j = [A_j, C_j]$ for some C_j in Y, whenever A_j is in V, $1 \leqslant j \leqslant n$.[11]

(b) If $A \to A_1 A_2 \cdots A_n$ is the i-th production in P, $A \neq S$, then for C in Y, $[A, C] \to B_1 B_2 \cdots B_n$ is in P', where $B_j = A_j$ if A_j is in Σ, and $B_j = [A_j, C_j]$ if A_j is in V, $1 \leqslant j \leqslant n$. However, the string $C_1 C_2 \cdots C_n$, where $C_j = \epsilon$ if A_j is in Σ, must be in $\mu(i, C)$.

If the above conditions hold, then we say G' is the *convolution* of G with Y and μ. Call μ a *uniquely invertible* function on G and Y.

Informally, the grammar G' is the grammar G with certain information (the elements of Y) carried at each node of its parse trees, with the exception of the root and the leaves. The information is passed from a node to its ancestor. Because of rule (2), the information is such that given a parse tree in G, one can find a unique parse tree in G' with the same yield. This is shown in the next lemma.

LEMMA 5.4. *Let $G = (V, \Sigma, P, S)$ be a proper* CFG, *Y a finite set and μ a uniquely invertible function on G and Y. Let G' be the convolution of G with Y and μ. Let h be the homomorphism defined by $h([A, C]) = A$ for all A in $V - \{S\}$, C in Y and $h(A) = A$ for all A in $\Sigma \cup \{S\}$, and extend h to productions of G' so that if j is a production of G', then $h(j)$ is the number*

[11] Note that if there are identical productions in P, a new set is made for each production.

of the production of G from which j was constructed according to rule (a) *or* (b) *above. Then h is a* 1–1 *tree correspondence from* G' *to* G.

Proof. Under the correspondence h, any tree in G' becomes a tree in G. Let D be a parse tree in G. We will uniquely assign new labels to the nodes of D to form a new tree D' which will be a parse tree in G' with the same yield as D. The assignment of labels to nodes will be by induction on the height of a node.

Let N be a node (not the root) of D with label i, and suppose all of N's descendants are leaves. Then $\mu(i, C) = \{\epsilon\}$ for some one value of C in Y and $\mu(i, B) = \varphi$ for $B \neq C$. Thus, if the i-th production of P is $A \rightarrow w$, change the label of node N to the corresponding production $[A, C] \rightarrow w$.

Now, let N be a node (not the root) all of whose descendants are either leaves or have had new labels assigned. Let the label at node N be the i-th production of P, $A \rightarrow B_1B_2 \cdots B_n$. For $1 \leqslant j \leqslant n$, define $C_j = \epsilon$ if B_j is in Σ and $C_j = C$ if the new label of the j-th descendant of N is of the form $[B_j, C] \rightarrow \alpha$. Let C be the unique element of Y such that $C_1C_2 \cdots C_n$ is in $\mu(i, C)$. Then the new label of N is $[A, C] \rightarrow D_1D_2 \cdots D_n$, where $D_j = B_j$ if B_j is in Σ, and $D_j = [B_j, C_j]$ if B_j is in V.

If the label of the root is $S \rightarrow B_1B_2 \cdots B_n$, and all its descendants are leaves or have new labels, change the label of the root to $S \rightarrow D_1D_2 \cdots D_n$, where the D_j's are defined from the B_j's as in the paragraph above.

The above assignment produces the unique parse tree in G' whose yield is the same as that of G. Thus, h is a 1–1 tree correspondence.

Our next task is to extend the results on proliferation rate to results on output growth. We do this by modifying a GSDT so that each translation symbol used actually produces arbitrarily long outputs.

LEMMA 5.5. *Given a* GSDT $F = (G, \Delta, \Gamma, R)$, $G = (V, \Sigma, P, S)$, *there is a* GSDT $F' = (G', \Delta, \Gamma', R')$, $G' = (V', \Sigma, P', S)$, *with* $T(F') = T(F)$, *such that if* A_i *is in* Γ', $A \neq S$, *then there is no constant upper bound on the length of a string* $v(A_i)$ *which can be defined at a node of a tree in* G'. *Moreover, there exists a* 1–1 *tree correspondence from* G' *to* G.

Proof. Let Γ_1 be the set of A_i in Γ, $A \neq S$, such that there is a finite least upper bound on the length of $v(A_i)$ defined at any node of any tree in G, and let b be the maximum of these bounds.[12] F' will use only those translation symbols in $\Gamma - \Gamma_1$. When a rule involves a symbol A_i in Γ_1, that symbol will be replaced by one of the values $v(A_i)$ would assume if

[12] We leave it to the reader to show that it is decidable if A_i is in Γ_1. Decidability of this question is not, however, needed in the proof of Lemma 5.5.

defined at some node N. The grammar G and the rules must then be modified to insure that whatever is subsequently derived from node N will cause the selected value of $v(A_i)$ to actually occur at N.

Let Y be the set of maps from Γ_1 to $\bigcup_{j=0}^{b} \Delta^j$. Now, suppose production i in P is of the form $A \rightarrow w_0 A_1 w_1 A_2 w_2 \cdots A_m w_m$, where the A's are in V, the w's in Σ^*. Let $M, M_1, M_2, \ldots, M_m$ be in Y.

For each $\tau_j(A)$ [13] in Γ_1, let $\tau_j(A) = \alpha_j$ be the rule associated with production i by R. Define the function μ such that $M_1 M_2 \cdots M_m$ is in $\mu(i, M)$ if and only if:

(1) The string $M(\tau_j(A))$ is the string formed by replacing each instance of $\tau_k(A_l)$ in α_j by $M_l(\tau_k(A_l))$. (Obviously, $\tau_k(A_l)$ is in Γ_1.)

(2) $M(\tau_j(B)) = \epsilon$ if $B \neq A$.

μ is uniquely invertible on G and Y. Let the grammar $G' = (V', \Sigma, P', S)$ be the convolution of G with Y and μ. Then, the homomorphism h such that $h(S) = S$ and $h([A, M]) = A$ for all M in Y is a 1–1 tree correspondence from G' to G.

If a node of a tree in G' is labeled $[A, M] \rightarrow \alpha$, then $M(\tau_i(A))$ is the value of $v(\tau_i(A))$ defined by F at the corresponding node of the corresponding tree in G. This is easy to verify by induction on the height of a node.

F' is constructed as follows. Let $\Gamma' = \{S_1\} \cup \{\tau_i([A, M]) \mid \tau_i(A)$ is in $\Gamma - \Gamma_1$, M in $Y.\}$ Define R' in the following manner:

(1) Suppose R associates $\tau_i(A) = B_1 B_2 \cdots B_m$ with production $A \rightarrow A_1 A_2 \cdots A_n$, $A \neq S$. Let $[A, M] \rightarrow C_1 C_2 \cdots C_n$ be a corresponding production in P', where $C_j = A_j$ if A_j is in Σ and $C_j = [A_j, M_j]$ otherwise. Then R' associates $\tau_i([A, M]) = D_1 D_2 \cdots D_m$ with $[A, M] \rightarrow C_1 C_2 \cdots C_n$, where for $1 \leqslant j \leqslant m$:

(i) $D_j = B_j$ if B_j is in Δ.
(ii) $D_j = \tau_k([A_l, M_l])$ if $B_j = \tau_k(A_l)$ is in $\Gamma - \Gamma_1$,
(iii) $D_j = M_l(B_j)$ if $B_j = \tau_k(A_l)$ is in Γ_1.

(2) If R associates $S_1 = B_1 B_2 \cdots B_m$ with $S \rightarrow A_1 A_2 \cdots A_n$, then R' associates $S_1 = D_1 D_2 \cdots D_m$ with each $S \rightarrow C_1 C_2 \cdots C_n$ in P', where $C_1, C_2, \ldots, C_n$ and $D_1, D_2, \ldots, D_m$ are related as above.

A straightforward argument by induction on the height of a node shows that $T(F') = T(F)$.

[13] Note the change in notation for symbol in Γ.

A GSDT satisfying Lemma 5.5 will be called *reduced*.

We now prove another type of "pumping lemma," this time concerned with the length of output string, rather than proliferation rate.

LEMMA 5.6. *Let F be a reduced* GSDT (G, Δ, Γ, R) *with* $G = (V, \Sigma, P, S)$. *Let A_i be in Γ, $A \neq S$. Then there are constants c_1 and c_2 such that for all m there is a tree in grammar G, with root labeled by an A-production and $c_1 + c_2 m$ nodes, such that for $v(A_i)$ defined at the root of this tree, $|v(A_i)| \geqslant m$.*

Proof. Let Γ have s symbols, V have t symbols, and let r be the maximum length of the right side of a rule. Since F is reduced, there is some tree D, whose root is labeled by an A-production, and for which $|v(A_i)| \geqslant (st + 2)^r$, when $v(A_i)$ is defined at the root. It is straightforward to find a path Π in D, say $\Pi = N_1, N_2, \ldots, N_k$, $k > st + 2$, such that N_1 is the root, and N_k a leaf. For each N_j in Π, there is some string $v(B_j)$ defined at N_j, with $|v(B_j)| \geqslant |v(B_{j+1})| \geqslant |v(B_j)|/r$, and $B_1 = A_i$; moreover, for $1 \leqslant j < k$, the rule $B_j = \alpha$ applied at node N_j is such that α has at least one instance of B_{j+1}.

Since $v(B_{k-1})$ at N_{k-1} has length at most r, there are at least $st + 1$ values of j for which $|v(B_j)| > |v(B_{j+1})|$, where $v(B_j)$ and $v(B_{j+i})$ are defined at nodes N_j and N_{j+1}, respectively. Thus we can find p and q such that $p > q$, $B_p = B_q$ and at the appropriate nodes, $|v(B_p)| < |v(B_q)|$. Let c_2 be the number of nodes in the subtree D_q which has root N_q, exclusive of the subtree D_p which has root N_p. Let c_1 be the number of remaining nodes in tree D. Form the sequence of trees $E_1, E_2, \ldots, E_m, \ldots$, where $E_1 = D_q$, and E_j is formed from E_{j-1} by replacing the subtree D_p by D_q. Then $v(B_q)$ defined at the root of E_m has length at least m. Therefore, if E_m replaces D_q in D, the length of $v(A_i)$ at the root of this tree is at least m. Moreover, this tree has $c_1 + c_2 m$ nodes.

LEMMA 5.7. *Let* $F = (G, \Delta, \Gamma, R)$, $G = (V, \Sigma, P, S)$, *be a reduced* GSDT. *If the proliferation rate of S_1 is at least $c'n^p$, $p \geqslant 0$, $c' > 0$, then there is a constant $c > 0$ and an infinite set of parse trees in grammar G, such that for each tree in the set with n nodes, $v(S_1)$ has length at least cn^{p+1}.*

Proof. By Lemma 5.6, there is a constant c_3 such that for any A_i in Γ, $A \neq S$, and any $j \geqslant 1$, there is a subtree D in grammar G, for which $v(A_i)$ defined at the root of D has length at least j, and D has no more than $c_3 j$ nodes. (Let c_3 be twice the largest of the c_1's and c_2's defined in Lemma 5.6.)

We also observe the following. Let D_1 be a tree with a path Π of length m. Construct tree D_2 by replacing each node N which either is not on path Π

or has a nonterminal label by the shortest tree with terminal leaves whose root has the same label as N. There is a constant c_4, depending only on G, such that D_2 has no more than $c_4 m$ nodes.

Now, let n be an arbitrary integer greater than zero, $j = [n/2c_3]$ and $m = n - c_3 j$. By Lemma 4.4, there exists α in $\mathscr{H}_F{}^*$ such that $|\alpha(S_1)| \geqslant c'm^p$. Construct a parse tree with a path Π represented by α, having at most $c_4 m$ nodes. Let Γ have s symbols. Then $\alpha(S_1)$ has at least $c'm^p/s$ instances of some symbol B_k. Since G is assumed to be proper and $m > 0$, $B \neq S$. By Lemma 5.6, we can replace the last node of the path Π by a subtree with at most $c_3 j$ nodes, such that $v(B_k)$ defined at the root of the subtree has length at least j.

Thus, $v(S_1)$ defined at the root of the entire tree has length at least $jc'm^p/s$. Also, the number of nodes of the entire tree is at most $c_4 m + c_3 j$ nodes. By observing that for large n, both j and m are bounded above and below by positive multiples of n, we have the desired result.

LEMMA 5.8. *Let F be a reduced* GSDT *as in Lemma* 5.7. *If the proliferation rate of S_1 is $f(n) \geqslant [(c')^n]$, then there is a constant $c > 1$ such that for an infinite set of parse trees, each tree in the set with n nodes has $|v(S_1)| \geqslant c^n$, whenever $v(S_1)$ is defined.*

Proof. Similar to Lemma 5.7.

THEOREM 5.1. *Let T be defined by a* GSDT. *Then T is defined by some* GSDT *$F = (G, \Delta, \Gamma, R)$, with $G = (V, \Sigma, P, S)$, such that one of three cases holds:*

(1) *There is a constant c such that if (x, y) is in T, then $|y| \leqslant c$.*

(2) *There are positive constants c_1 and c_2, and an integer $i \geqslant 1$ such that if $v(S_1)$ is defined at the root of a parse tree with n nodes, then $|v(S_1)| \leqslant c_2 n^i$, and there is an infinite set of parse trees in G such that $v(S_1)$ defined at the root of a tree with n nodes has length at least $c_1 n^i$.*

(3) *There are constants $c_1 > 1$ and $c_2 > 1$ such that if $v(S_1)$ is defined at the root of a parse tree with n nodes, then $|v(S_1)| \leqslant (c_2)^n$, and there is an infinite set of parse trees such that $v(S_1)$ defined at the root of a tree with n nodes has length at least $(c_1)^n$.*

Proof. Let F be reduced. Assume (1) does not hold, and let $f(n)$ be the proliferation rate of S_1. Now $f(n) \neq 0$ for any n. (For if $f(n) = 0$, then $f(n') = 0$ for all $n' \geqslant n$, and (1) could be shown.) Suppose $v(S_1)$ is in $\Gamma^{(j)}$, $j \geqslant 0$. By the foregoing (for $j = 0$), and Theorem 4.1, $f(n) \geqslant c_3 n^j$ for some $c_3 > 0$. Case (2) then follows from Lemmas 5.2 and 5.7 with $i = j + 1$. If S_1 is in no $\Gamma^{(j)}$, case (3) follows from Theorem 4.6 and Lemmas 5.3 and 5.8.

THEOREM 5.2. *Let T be defined by an unambiguous* GSDT. *Then one of three cases holds*:

(1) *There is a constant c such that for all (x, y) in T, $|y| \leqslant c$.*

(2) *There are positive constants c_1 and c_2 and a positive integer i such that if (x, y) is in T, then $|y| \leqslant c_2(|x| + 1)^i$, and for an infinity of x there exists (x, y) in T such that $|y| \geqslant c_1(|x| + 1)^i$.*

(3) *There are constants $c_1 > 1$ and $c_2 > 1$ such that if (x, y) is in T, then $|y| \leqslant (c_2)^{(|x|+1)}$, and for an infinity of x there exists (x, y) in T such that $|y| \geqslant (c_1)^{(|x|+1)}$.*

Proof. This result follows from Theorem 5.1 and the observation that for an unambiguous CFG G, there is a constant c such that the parse tree for each x in $L(G)$ has at most $c(|x| + 1)$ nodes.

THEOREM 5.3. *Let $F = (G, \Delta, \Gamma, R)$ be a* GSDT, *with $G = (V, \Sigma, P, S)$. If S_1 is in $\Gamma^{(i)}$, then there is a constant c such that for all x in $L(G)$, there exists (x, y) in $T(F)$, with $|y| \leqslant c(|x| + 1)^{i+1}$.*

Proof. It is left to the reader to show that there is a constant c_1, such that every x in $L(G)$ is the yield of a parse tree with at most $c_1(|x| + 1)$ nodes. (Essentially, given any tree with yield x, one can modify it to eliminate large subtrees with ϵ yield and long sequences of nodes with a single descendant. These modifications produce the desired tree with yield x.) By Lemma 5.2, for some constant c_2, the translation produced by such a tree is of length at most $c_2 c_1(|x| + 1)^{i+1}$.

THEOREM 5.4. *For an arbitrary* GSDT *F, there is a constant $c > 1$ such that for each x in the domain of $T(F)$, there exists (x, y) in $T(F)$ and $|y| \leqslant c^{(|x|+1)}$.*

Proof. Similar to Theorem 5.3.

VI. TREE AUTOMATA

In this section we develop an exact characterization for translations defined by a certain class of GSDT's in terms of finite automata operating upon the parse trees of context free grammars. This type of finite automaton we call a tree automaton. Recently, some interest has been focused upon finite automata with the domain of definition extended to directed graphs, especially trees [17–20]. However, our notion of tree automata differs in substantial respects from the various notions of [17–20].[14]

Intuitively, our tree automaton is a deterministic finite transducer operating on a parse tree of a CFG $G = (V, \Sigma, P, S)$. A tree automaton A defines translations of an input word x in $L(G)$ in the following manner. Let D be a parse tree with yield x. A is initially in its start state q_0 and is at the root of the parse tree. A then executes a sequence of moves. A move is determined by the label of the node N at which A is positioned and the current state of A. In one move, A changes state, emits a finite length output string, and moves either to the ancestor of node N, a designated descendant of node N or remains at node N. If A can make some sequence of moves on D, during which it emits the output strings $y_1, y_2, \ldots, y_n$ (in that order), such that it begins this sequence of moves on the root in state q_0 and halts on the root in the final state, then $y_1 y_2 \cdots y_n$ is said to be a translation of x.

Formally, a *tree automaton* is a 6-tuple $A = (Q, G, \Delta, \delta, q_0, q_f)$ where:

(1) Q is a finite set of *states*.

(2) $G = (V, \Sigma, P, S)$ is a proper CFG, called the *underlying* grammar.

(3) Δ is a finite set of *output symbols*.

(4) δ is a mapping from $(Q - \{q_f\}) \times (P \cup \Sigma \cup \{\epsilon\})$ to $Q \times I \times \Delta^* \cup \varphi$, where $I = \{-1, 0, 1, 2, \ldots, p\}$. p is the maximum length of the right side of a production in P. If $\delta(q, L) = (q', i, x)$, and L is an S-production, then $i \neq -1$. If L is in $\Sigma \cup \{\epsilon\}$, then $i \leqslant 0$. If L is a production with r symbols on the right, $r \geqslant 1$, then $i \leqslant r$. If L is a production of the form $A \to \epsilon$, then $i \leqslant 1$. (These conditions ensure that A will always have a node to move to).

(5) q_0, in Q, is the *start state*.

(6) q_f, in Q, is the *final state*.

We describe the action of A on a parse tree D with yield x, by defining three functions of time (number of moves made), $s(t)$, $N(t)$ and $\theta(t)$. $s(t)$ is the state of A after t moves. $N(t)$ is the node at which the automaton is positioned after t moves, and $\theta(t)$ is the accumulated output after t moves. $s(0) = q_0$. $N(0)$ is the root of D and $\theta(0) = \epsilon$. Inductively, suppose $s(t)$, $N(t)$ and $\theta(t)$ have been defined.

[14] Since this was written, interest in our model has been stimulated. In particular, M. O. Rabin has recently shown that they are equivalent in their tree recognizing ability to the automaton of [17].

If $s(t) = q_f$ and $N(t)$ is the root, then $\theta(t)$ is a *translation* of x. $s(t')$, $N(t')$ and $\theta(t')$ are undefined for $t' > t$.

If $s(t) \neq q_f$, let the label of node $N(t)$ be L in $P \cup \Sigma \cup \{\epsilon\}$. If $\delta(s(t), L) = (q, i, x)$, then $\theta(t+1) = \theta(t)x$ and $s(t+1) = q$. $N(t+1)$ is $N(t)$, the ancestor of $N(t)$ or the i-th descendant of $N(t)$ from the left, as $i = 0$, $i = -1$ or $i > 0$, respectively.

The *translation defined by* A, denoted $T(A)$, is the set of (x, y) such that y is a translation of x. If T is $T(A)$ for some tree automaton A, then T will be called a *tree automaton translation* (TAT).

VII. Tree Automata and GSDT's

We shall show that the class of tree automaton translations is exactly the class of translations defined by GSDT's which have a linear relation between the size of tree and length of output. The argument proceeds by a series of lemmas.

Lemma 7.1. *Let* $A = (Q, G, \Delta, \delta, q_0, q_f)$ *be a tree automaton. Then there is a constant* c *such that on any tree* D *with* n *nodes, if* $s(t) = q_f$, *then* $t \leqslant cn$.

Proof. Let c be the number of elements in Q. If $s(t)$ is defined for some $t > cn$, then there exist t_1 and t_2, such that $s(t_1) = s(t_2)$ and $N(t_1) = N(t_2)$. It should be clear that A is in a loop; that is, for all i, $s(t_1 + i(t_2 - t_1)) = s(t_1)$. Thus, A can never enter state q_f if started on the root of tree D.

Lemma 7.2. *If* $T = T(A')$, *for some tree automaton*

$$A' = (Q', G', \Delta, \delta', q_0, q_f),$$

then $T = T(A)$, *where* $A = (Q, G, \Delta, \delta, q_0, q_f)$ *is a tree automaton which, if it moves from a node* N *of some parse tree to one of* N*'s descendants, will always return to* N. *There is a* 1–1 *tree correspondence from* G *to* G'.

Proof. Let $G' = (V', \Sigma, P', S)$. The gist of the argument is that we may augment the grammar G' to incorporate at each node N (except for leaves and the root) information that answers the question: "If A' reaches node N in state q, will A' eventually reach the ancestor of N, and if so, in what state?"

To that aim, let Y be the set of maps from $Q - \{q_f\}$ to $Q \cup \{\varphi\}$. We define μ, a mapping from $P \times Y$ to finite subsets of Y^*.

Let the i-th production of G be $B \to B_1 B_2 \cdots B_m$ and let M_j, $1 \leqslant j \leqslant m$, be ϵ or an element of Y, as B_j is or is not a terminal. We define M, the unique element of Y such that $\mu(i, M)$ contains $M_1 M_2 \cdots M_m$, as follows. For each q in $Q - \{q_f\}$, we define a sequence of states $q_1, q_2, \ldots$, by the following procedure:

(1) $q_1 = q$.

(2) If $q_j, j \geq 1$ has been defined and $\delta'(q_j, i) = (p, k, x)$ go to step 3, 4 or 5 as $k = 0$, $k = -1$ or $k > 0$.

(3) Set $q_{j+1} = p$. Go to step 6.

(4) Set $M(q) = p$. Halt.

(5) If B_k is not a terminal, and $M_k(p) = \varphi$, set $M(q) = \varphi$. Halt. If $M_k(p) = p'$, set $q_{j+1} = p'$ and go to step 6. If B_k is a terminal, determine if A', starting in state p at a node labeled B_k, will ever return to the ancestor of that node. If not, set $M(q) = \varphi$. Halt. If A' will return in state p', set $q_{j+1} = p'$ and go to step 6. If B_k is ϵ (in which case the i-th production is $B \to \epsilon$), perform the same computation as for the case in which B_k is a terminal.

(6) If q_{j+1} has just been defined and $q_{j+1} = q_f$, then set $M(q) = \varphi$. If $j + 1$ exceeds the number of states of A', set $M(q) = \varphi$. Otherwise, return to step 2.

Intuitively, based on the assumption that $M_1, M_2, \ldots, M_m$ correctly answer the question stated above for the descendants of a node N, then M will answer correctly for N. Let $G = (V, \Sigma, P, S)$ be the convolution of G' with Y and μ. Let h be the 1–1 tree correspondence from G to G' defined by $h(S) = S$ and $h([B, M]) = B$ for all B in $V' - \{S\}$ and M in Y.

Let D be a tree in G and $D' = h(D)$. We observe by induction on the height of a node N in D that if the label of N is $[B, M] \to \alpha$, and A' is started in state q, on the node N' of D' corresponding to N, then A' will return to the ancestor of N' if and only if $M(q) \neq \varphi$. If A' does return, $M(q)$ is its state at that time. Armed with this observation, it is straightforward to specify a tree automaton A which will make the same moves on a parse tree D in G as A' will make on the corresponding tree $h(D)$ in G', provided that A' defines a translation on $h(D)$.

Note, however, that for some parse trees $D' = h(D)$ in G', A' may halt on one of the descendants of the root or loop around the root and its descendants. In both cases, A is defined to make one move on D, staying on the root of D and halting in a nonfinal state. No translation is produced by A on D. These conditions can be detected by observing the production used at the root of D.

LEMMA 7.3. *If $T = T(A)$ for some tree automaton $A = (Q, G, \Delta, \delta, q_0, q_f)$ where $G = (V, \Sigma, P, S)$, then $T = T(A_1)$ for a tree automaton $A_1 =$*

$(Q, G_1, \Delta, \delta, q_0, q_f)$ such that A_1 produces a translation on every parse tree of G_1. If G is unambiguous, then so is G_1.

Proof. We may assume A to have been constructed in Lemma 7.2. Whether A will produce a translation on a given tree is determined solely by the S-production labeling the root of the tree. For each S-production $S \to \alpha$ for which A produces no translation on trees with $S \to \alpha$ as label of the root, we can delete $S \to \alpha$ from P without altering the translation defined by A. Let $G_1 = (V, \Sigma, P_1, S)$ be the grammar with all such productions removed and let $A_1 = (Q, G_1, \Delta, \delta, q_0, q_f)$. Then A_1 produces a translation on every parse tree in grammar G_1 with yield in Σ^*.

We say a GSDT $F = (G, \Delta, \Gamma, R)$ is *linear* if there is a constant c such that every parse tree in grammar G with n nodes produces a translation of length at most cn.

LEMMA 7.4. *Let $F = ((V, \Sigma, P, S), \Delta, \Gamma, R)$ be a reduced* GSDT. *Then F is linear if and only if S_1 is in $\Gamma^{(0)}$.*

Proof. The "if" portion follows from Lemma 5.2. For the "only if" part, observe that if S_1 is not in $\Gamma^{(0)}$, then, by Lemma 4.5, there is a positive constant c_1 such that the proliferation rate of S_1 is at least $c_1 n$. By Lemma 5.7, there is a positive constant c_2 and an infinite set of trees such that a tree with n nodes produces a translation of length at least $c_2 n^2$.

LEMMA 7.5. *If T is a* TAT, *then T is defined by a linear* GSDT.

Proof. Without loss of generality we can assume T is $T(A_1)$ where A_1 is the tree automaton $(Q, G_1, \Delta, \delta, q_0, q_f)$ of Lemma 7.3 and $G_1 = (V, \Sigma, P_1, S)$.

Let F be the GSDT (G, Δ, Γ, R) where $\Gamma = \{\tau_1(S)\} \cup \{\tau_q([C, M]) \mid q$ is in Q, $[C, M]$ is in $V - \{S\}$, and $M(q) \neq \varphi\}$.[15]

The symbol $\tau_q([C, M])$ is intended to represent the output of A_1 when started in state q on a node whose label is a $[C, M]$-production, until the time that A_1 moves to the ancestor of that node. We define the rules of F so that $v(\tau_q[C, M])$ defined at any node will in fact be the string desired.

Let $[C, M] \to B_1 B_2 \cdots B_m$ be the p-th production of G_1, with $B_j = [C_j, M_j]$, if B_j is in V. We construct the rule $\tau_q([C, M]) = \alpha$ associated with this production by defining a sequence of strings $\alpha_1, \alpha_2, \ldots$ in $(\Gamma \cup \Delta)^*$ and a sequence of states $q_1, q_2, \ldots$ in Q. Initially, $\alpha_1 = \epsilon$ and $q_1 = q$. Suppose α_j and q_j have been defined, $j \geqslant 1$. Let $\delta(q_j, p) = (q', k, x)$. Four cases arise:

[15] We assume that states of A_1 are identified with integers.

(1) If $k = 0$, then $q_{j+1} = q'$ and $\alpha_{j+1} = \alpha_j x$.

(2) If $k = -1$, then α_{j+1} and q_{j+1} are not defined. Instead, $\alpha = \alpha_j x$, and the process terminates.

(3) If $k > 0$ and B_k is in V, then $\alpha_{j+1} = \alpha_j x \tau_{q'}(B_k)$ and $q_{j+1} = M_k(q')$.

(4) If $k > 0$ and B_k is in Σ, let y be the string of output symbols A_1 will produce up to the time it returns to the original node, and let q'' be the state at that time. Then $\alpha_{j+1} = \alpha_j xy$ and $q_{j+1} = q''$.

Since $\tau_q([C, M])$ is only defined if $M(q) \neq \varphi$, the above process must terminate (i.e., (2) becomes applicable.)

Next, let $S \to B_1 B_2 \cdots B_m$ be the p-th production. Associate with this production the rule $\tau_1(S) = \alpha$, where α is constructed from $B_1, B_2, \ldots, B_m$ by constructing a sequence of strings $\alpha_1, \alpha_2, \ldots$ and sequence of states $q_1, q_2, \ldots$, in a manner similar to the above process. Let $B_j = [C_j, M_j]$ if B_j is in V. Initially, $\alpha_1 = \epsilon$ and $q_1 = q_0$, the start state of A_1. Suppose α_j and q_j have been defined, $j \geqslant 1$. Let $\delta(q_j, p) = (q', k, x)$. Case (2) above $(k = -1)$ cannot arise, but in each of the other three cases above, α_{j+1} and q_{j+1} can be constructed by the procedures given in Cases (1), (3) and (4). When some $q_{j+1} = q_f$, the process terminates and $\alpha = \alpha_{j+1}$. We have constructed G_1 so that this event will always occur.

It is straightforward to show by induction on the height of a node N that if $v(\tau_q(B))$ is defined at N, then $v(\tau_q(B))$ is the output string A would produce if started at node N, in state q, up to the time A first moves to the ancestor of N. It then follows that $v(S_1)$ is the output of A when started at the root in state q_0, until A enters state q_f at the root. Lemma 7.1 implies that F is linear.

Combining Lemmas 7.4 and noting that every translation defined by a linear GSDT is defined by a reduced linear GSDT, we have the following result.

THEOREM 7.1. *If T is a* TAT, *then $T = T(F)$ for some* GSDT *$F = (G, \Delta, \Gamma, R)$, with $G = (V, \Sigma, P, S)$, such that S_1 is in $\Gamma^{(0)}$. If the grammar of the* TAT *is unambiguous, so is G.*

We can also prove the converse of Theorem 7.1. The construction is somewhat involved, so we begin with an informal description. Let $F = (G, \Delta, \Gamma, R)$ be a GSDT, with $G = (V, \Sigma, P, S)$, and assume that S_1 is in $\Gamma^{(0)}$. G must be modified so that at any node N of a parse tree, if α is the element of $\mathscr{H}_F{}^*$ representing the path from the root to N, then the string $\alpha(S_1)$ is available at N. Call this string the "key". The fact that there is a bound on the length of such a string will make the modification possible.

Let G_1 be the modified grammar. We will construct A to walk on the trees of G_1. If N is a node of a tree, then when A reaches N from its ancestor it will have in its finite control, a "pointer" set to one of the symbols of the key. This symbol will represent the name of the translation symbol, the value of which A is about to produce at its output. Let C be the symbol to which the pointer is set. The semantic rule for the computation of the value of C at this node involves various symbols of Γ. These elements are each found in the key associated with the semantic rule at a particular descendant of N. For each of these symbols, in turn, A moves to the proper descendant of N, setting the pointer to the correct position of the new key.

The essential step in showing that A can operate correctly is concerned with what occurs when A returns to node N with a pointer to a position in the key at one of N's descendants. By comparison of this key with the key at N, A can determine for what position of the key at N it was attempting to produce a translation string. Let C be the symbol at this position. A may have just made an excursion to a descendant corresponding to the last symbol (in Γ) of the semantic rule for C. If so, A records the key at N in its finite control, sets a pointer to the proper instance of C in that key and moves to the ancestor of N. If in the rule for C there is a symbol in Γ following the one for which the excursion was made, A moves to the proper descendant of N, setting a pointer to that symbol.

We will now formalize the above argument.

THEOREM 7.2. *If T is $T(F)$, for a* GSDT $F = (G, \Delta, \Gamma, R)$ *with* $G = (V, \Sigma, P, S)$, *and* S_1 *in* $\Gamma^{(0)}$, *then T is a* TAT. *If G is unambiguous, then so is the underlying grammar of the tree automaton.*

Proof. Let b be the maximum $|\alpha(S_1)|$, for α in $\mathscr{H}_F{}^*$. We construct the CFG $G_1 = (V_1, \Sigma, P_1, [S, S_1])$, where $V_1 = V \times \bigcup_{i=0}^{b} \Gamma^i$. Let $B \to B_1 B_2 \cdots B_m$ be the p-th production in P. Let w be in Γ^*, $|w| \leqslant b$. The production $[B, w] \to C_1 C_2 \cdots C_m$, where $C_j = B_j$ if B_j is in Σ and $C_j = [B_j, h_{pB_j}(w)]$ otherwise, is in P_1, if $|h_{pB_j}(w)| \leqslant b$ for all h_{pB_j} in $\mathscr{H}$ and all appropriate j. Give this production the "number" (p, w) in P_1. The homomorphism h given by $h([B, w]) = B$, for B in V, $h(B) = B$, for $B \in \Sigma$, and $h((p, w)) = p$ is a 1–1 tree correspondence from G_1 to G.

Define A to be the tree automaton $(Q, G_1, \Delta, \delta, q_0, q_f)$, where $Q = \{q_f\} \cup \{[X, w, i] \mid w \text{ in } \Gamma^*, |w| \leqslant b, 1 \leqslant i \leqslant |w|, X \text{ in } \{D, U\}\}$. $q_0 = [D, S_1, 1]$. (The symbols D and U in the state of A indicate whether A has just moved down (D) or up (U) the tree. Initially, A acts as though it had just moved down, to the root. The second component stores a key, the third is the pointer.)

Before describing the moves of A, we must introduce some notation. Let (p, y) be a production in P_1 and let h_{pB} be in $\mathscr{H}_F$. We can write $y = C_1C_2 \cdots C_n$, where the C's are in Γ, and we can write $h_{pB}(y) = y_1y_2 \cdots y_n$, where $h_{pB}(C_j) = y_j = D_{j1}D_{j2} \cdots D_{jr_j}$ with D_{ji} in Γ, $1 \leqslant j \leqslant n$. If D_{ji} in y_j is the k-th symbol of $h_{pB}(y)$, then the *image* of the k-th position of $h_{pB}(y)$ in y is j and the *subposition* of the k-th position of $h_{pB}(y)$ is i. That is, the image of a position in $h_{pB}(y)$ is the position of y which produces it, and the subposition is its relative position among the positions produced by its image. Let Γ_B be the set of elements of Γ which are of the form B_j for some j. We shall call a production (p, w) of P_1 a *B-production* if production p of P is a B-production.

Let $[D, w, i]$ be a state of A, and B_j the i-th symbol of w. If A finds itself in this state, at some node N, then w will be the string in Γ^* associated with the left side of the production labeling N. Let this production be (p, w), and let $B_j = B_1B_2 \cdots B_m$ be the rule which R associates with the p-th production of P. A must next emit the string $v(B_j)$ defined at node N, then move to the ancestor of node N. If $B_1B_2 \cdots B_m$ is in Δ^*, then

$$\text{(R1)} \qquad \delta([D, w, i], (p, w)) = ([U, w, i], -1, B_1B_2 \cdots B_m).$$

That is, A emits $B_1B_2 \cdots B_m$ and returns to the ancestor of N, its job done. (Note that when A last left the ancestor of N, it entered state $[D, w, i]$ and that A returns in state $[U, w, i]$. This relation will be shown true in general after we have completed the specification of δ.)

If for some smallest s, B_s is in Γ, let B_s be in Γ_C, and let the k-th descendant of N be that descendant whose label is a C-production. Define $y = h_{pC}(w)$. Let position n of y be that position whose subposition is 1 and whose image is the i-th position of w. Then:

$$\text{(R2)} \qquad \delta([D, w, i], (p, w)) = ([D, y, n], k, B_1B_2 \cdots B_{s-1}).$$

(In explanation of (R2), it is possible that in order to compute $v(B_j)$ at node N, A must compute some number of translation strings at N's descendants. The rule for B_j is examined and the prefix of symbols in Δ is emitted. When the first symbol in Γ is encountered, A sets its state to indicate the desired translation and moves to the proper descendant.)

Now, let $[U, w, i]$ be a state of A. If A finds itself in this state at some node N, then A has just returned from one of the descendants of N—the unique descendant labeled by a production (q, w) in P_1, for some q. Let the label of node N be (p, y).

Then $w = h_{pB}(y)$ for some B in V. Let the image of the i-th position of w in y be l. Let production p of P be a C-production, and suppose that the l-th symbol of y is C_j for some j. Also, assume that R associates $C_j = D_1 D_2 \cdots D_m$ with production p. Let k be the subposition of the i-th position of w, and suppose that D_s is in Γ_B, and exactly $k - 1$ of $D_1, D_2, \ldots, D_{s-1}$ are in Γ_B. Then A has just emitted the portion of $v(C_j)$ corresponding to D_s. Three cases arise:

(1) If N is not the root (p is not an S-production), and all of $D_{s+1}, D_{s+2}, \ldots, D_m$ are in Δ, then:

$$\text{(R3)} \qquad \delta([U, w, i], (p, y)) = ([U, y, l], -1, D_{s+1} D_{s+2} \cdots D_m).$$

(In this case, A finishes emitting $v(C_j)$ at node N and returns to the ancestor of N.)

(2) If N is the root (p is an S-production) and all of $D_{s+1}, D_{s+2}, \ldots, D_m$ are in Δ, then

$$\text{(R4)} \qquad \delta([U, w, i], (p, y)) = (q_f, 0, D_{s+1} D_{s+2} \cdots D_m).$$

(Here, it must be that $j = 1$ and $C = S$. A finishes emitting $v(S_1)$, the correct translation, and ends its computation.)

(3) If for some smallest $t > s$, D_t is in Γ, let D_t be in Γ_E. Let r be the number of the unique descendant of N whose label is an E-production. Define $x = h_{pE}(y)$. Let n be the position of x whose image is position l of y, and whose subposition is equal to the number of $D_1, D_2, \ldots, D_t$ which are in Γ_E. Then:

$$\text{(R5)} \qquad \delta([U, w, i], (p, y)) = ([D, x, n], r, D_{s+1} D_{s+2} \cdots D_{t-1}).$$

We can show that $T(A) = T(F)$ by showing that:

(*) If on some parse tree in grammar G_1, A reaches a node N, other than the root, in state $[D, w, i]$, and the i-th symbol of w is B_j, then upon moving to the ancestor of N for the next time, A will enter state $[U, w, i]$, and the output of A from this time until A reaches the ancestor of N will be $v(B_j)$ defined at the node corresponding to N in the corresponding tree in grammar G.

The proof proceeds by induction on the height of node N.

The result is immediate for nodes of height 1 from (R1). Suppose it true for all descendants of node N. Let the label of node N be (p, w), and suppose that R associates the rule $B_j = C_1 C_2 \cdots C_m$ with production p.

If all of $C_1, C_2, \ldots, C_m$ are in Δ, (*) follows from (R1). Otherwise, let $C_{i_1}, C_{i_2}, \ldots, C_{i_s}$ be those of $C_1, C_2, \ldots, C_m$ in Γ. If A reaches node N in state $[D, w, i]$, and the i-th symbol of w is B_j, by (R2) A will emit $C_1C_2 \cdots C_{i_1-1}$. If C_{i_1} is in Γ_E, A will also, by (R2), move to the descendant of N whose label is a E-production. The state of A will be $[D, y, n]$, and the n-th symbol of y will be C_{i_1}. By $(*)$, A will emit $v(E_k)$, if C_{i_1} is the k-th translation symbol for E', and return to N in state $[U, y, n]$. Because of the way n is chosen in (R2), the value of i (in the original state $[D, w, i]$) is recovered by A. In a similar manner, by (R5), A continues to emit the portions of $v(B_j)$ corresponding to $C_{i_2}, \ldots, C_{i_s}$. Then, by (R3), A moves to the ancestor of N in state $[U, w, i]$ and emits the last symbols $C_{i_s+1}C_{i_s+2} \cdots C_m$ of $v(B_j)$.

From $(*)$, (R4) and an argument similar to the above for the case where N is the root, we conclude that $T(A) = T(F)$.

From Theorems 7.1 and 7.2, we have the following:

THEOREM 7.3. *A translation T is a* TAT *if and only if it is defined by a* GSDT $F = ((V, \Sigma, P, S), \Delta, \Gamma, R)$, *for which S_1 is in $\Gamma^{(0)}$. The underlying grammar of the* GSDT *may be made unambiguous if and only if the underlying grammar of F may be made unambiguous.*

VIII. CONCLUSIONS

We have investigated a class of translations called generalized syntax directed translations. They are effected by parsing the input according to a context free grammar and then defining various translation strings at the nodes, from the bottom up. We have shown that the function which relates the length of the output to the size of the input parse trees is either an integer power of the input length or exponential in the input length.

Next tree automata were defined, and it was shown that a translation is a tree automaton translation if and only if it is a GSDT with a linear relationship between the size of a tree and the output produced thereon.

On a theoretical level, we feel that there is a certain "naturalness" about both GSDT's at TAT's. For example, in both cases, there is an analog of the Chomsky normal form theorem. That is, the productions of the underlying grammar can be put in the form "nonterminal replaced by two nonterminals" or "nonterminal replaced by terminal or ϵ." The analogous statement is false for syntax directed translations [13]. It is expected that all the usual closure properties (composition with finite state mappings, for example) that hold for SDT's also hold for GSDT's and TAT's, with the exception of closure under inverse.

There may be some interesting characterizations of the GSDT's which are not TAT's, in terms of pebble automata [21, 22] walking on trees. We can show, at least in the case in which the underlying grammars are linear, that the GSDT's with S_1 in $\Gamma^{(i)}$ (in the usual meaning of these symbols) are equivalent to the translations produced by i pebble automata walking on trees of a CFG, under the constraint that the automaton must keep the pebble between itself and the root. We conjecture that this is true in general.

The range languages of the GSDT's and TAT's may form interesting classes. Their relation to some of the generalizations of context free languages, especially indexed languages [23], deserves attention. It is also possible that some of the common classes of languages, such as one-way, nondeterministic stack languages can be characterized in terms of pebble automata walking on trees. A hint of this possibility appears in [24]. Such an approach might lead to good proofs or new properties concerned with the theory of languages.

References

1. P. Naur (Ed.), Report on the algorithmic language ALGOL 60, *Comm. ACM* 3 (1960), 299–314.
2. E. T. Irons, A syntax directed compiler for ALGOL 60, *Comm. ACM* 4 (1961), 51–55.
3. J. Feldman and D. Gries, Translator writing systems, *Comm. ACM* 11 (1968), 77–113.
4. J. A. Feldman, A formal semantics for computer languages and its application in a compiler-compiler, *Comm. ACM* 9 (1968), 3–9.
5. R. M. McClure, TMG—a syntax-directed compiler, *Proc. ACM 20th Nat. Conf.* (1965), 262–274.
6. J. C. Reynolds, An introduction to the COGENT programming system, *Proc. ACM 20th Nat. Conf.* (1965), 422–436.
7. R. W. Floyd, On the nonexistence of a phrase structure grammar for ALGOL 60, *Comm. ACM* 5 (1962), 483–484.
8. R. E. Stearns and P. M. Lewis, Property grammars and table machines, *Information and Control* 14 (1969), 524–549
9. P. M. Lewis and R. E. Stearns, Syntax-directed transduction, *J. ACM* 15 (1968), 464–488.
10. K. Culik, Well translatable languages and ALGOL-like languages, *in Formal Language Description Languages* (T. Steele, Ed.), pp. 76–85, North Holland Press, Amsterdam, 1966.
11. D. H. Younger, Context free language processing in time n^3, *in Conference Record of 7th Annual Symposium on Switching and Automata Theory*, pp. 7–20, 1966.
12. A. V. Aho and J. D. Ullman, Properties of syntax directed translations, *J. Comp. Syst. Sci.* 3 (1969), 319–334.

13. A. V. AHO AND J. D. ULLMAN, Syntax directed translations and the pushdown assembler, *J. Comp. Syst. Sci.* 3 (1969), 37–56.
14. L. PETRONE, Syntax directed mappings of context free languages, *in Conference Record of 9th Annual Symposium on Switching and Automata Theory*, pp. 160–175, October 1968.
15. A. V. AHO AND J. D. ULLMAN, Characterizations and extensions of pushdown translations, *Math. Systems Theory* 5 (1971), 172–192.
16. D. E. KNUTH, Semantics of context free languages, *Math. Systems Theory* 2 (1968), 127–146; also see, *Math. Systems Theory* 5 (1971), 95–96.
17. J. W. THATCHER, Characterizing derivation trees of context free grammars through a generalization of finite automata theory, *J. Comp. Syst. Sci.* 1 (1967), 317–322.
18. J. DONER, "Decision Problems of Second-Order Logic," Technical report, System Development Corp., 1967. Santa Monica, California.
19. W. C. ROUNDS, Mappings and grammars on trees, *Math. Systems Theory* 4 (1970), 257–287.
20. M. A. ARBIB AND Y. GIVE'ON, Algebra automata 1: Parallel programming as a prolegomena to the categorical approach, *Information and Control* 12 (1968), 331–345.
21. M. O. RABIN, Mathematical theory of automata, *in Mathematical Aspects of Computer Science,* Proc. Symposia Applied Math., Vol. XIX, pp. 173–175, Amer. Math. Soc., Providence, RI, 1967.
22. M. J. FISCHER AND A. L. ROSENBERG, Limited random access Turing machines, *in Conference Record of 9th Annual Symposium on Switching and Automata Theory,* pp. 356–367, October 1968.
23. A. V. AHO, Indexed grammars—an extension of context free grammars, *J. ACM* 15 (1968), 647–671.
24. M. A. HARRISON AND M. SCHKOLNICK, A grammatical characterization of one-way nondeterministic stack languages, *J. ACM* 18 (1971), 148–172.